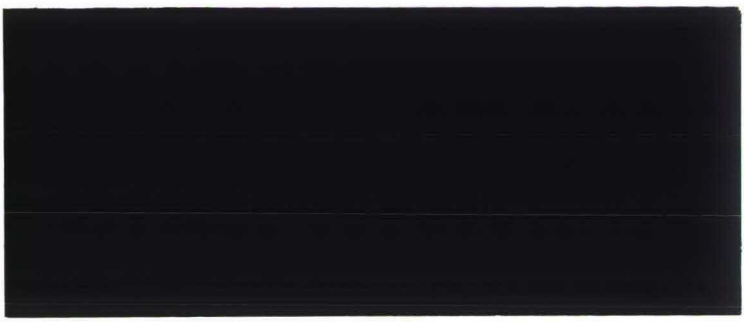
Infogame

Casimir, R.J.

Publication date:
1989



DEPARTMENT OF ECONOMICS

RESEARCH MEMORANDUM

INFOGAME
FINAL REPORT

Rommert J. Casimir

FEW 374

INFOGAME

FINAL REPORT

January 1989

Rommert J. Casimir
Tilburg University
P.O. Box 90153
5000LE Tilburg The Netherlands
R974CASI@HTIKUB5.BITNET

1	Introduction	3
2	Global game description	5
2.1	The infogame world	5
2.2	Time scale	6
2.3	Instruction by rules	10
2.4	Selective reporting	11
3	The game model in detail	12
3.1	An overview	12
3.2	Consumer model	13
3.3	Labor model	23
3.4	Machine model	34
3.5	Supply model	36
3.6	Production model	37
3.7	Finance model	41
3.8	Operational accounting	43
4	Implementation	46
4.1	Programming language and environment	46
4.2	Program structure	46
4.3	Input data	48
4.4	The intermediate file	50
4.5	Data structure	53

1 INTRODUCTION

Infogame* is a management game for research and education in information systems. A provisional description of the model was given in [6]; this report describes the model as it is currently implemented. A description for users is given in the a preliminary users' manual [8]. In November 1988, it has been decided to stop the Infogame research project; the reasons for its failure will be published elsewhere. Consequently no further versions of the program and the manual are planned. The primary purpose of the present paper is to document the design of Infogame; it also explains the mechanisms of the game to game administrators. It is certainly not intended for players. As the report ranges over a number of subjects (e.g. simulation modelling, operations research, information systems, and software engineering), topics of interest will be published separately. The organization of this report is as follows: section 2 and 3 describe the game from a modelling point of view; section 2 describes the main features, section 3 describes the model in detail. The implementation is described in section 4 from a software engineering point of view.

Acknowledgement: *I want to thank prof. Jack P. Kleijnen for his continuing support and encouragement during the development of Infogame.*

Differences with the preliminary design

Standard development methods, such as ISAC [17], assert that physical design of a computerized system should start only after the logical design has been finished; in contrast, in prototyping [2] a working program is considered an indispensable tool to establish user needs.

* Originally, the name "Infolab" has been used for the game, but this name was already in use for another project.

We did not use prototyping as it is usually defined, because actual users were not involved in the design, but we made a number of changes in the specifications with the user in mind, for example because is was difficult to clearly describe some features in the users' manual. Other changes were made because the original specifications were difficult to implement. The main differences between the design described in this report and the original specifications [6] are:

1) Technical progress is not implemented. All technologies, machine types, and materials are available at the start of play.

2) The notion of a "production line", embodying a technology and uniting a set of resources, is discarded. In the present version, resources are assembled for individual jobs. According to the specifications, the player had to set up production lines and allocate resources to production lines; in the present version, such commands are no longer needed. An additional advantage is that the present production system more closely resembles the IIasa model [1].

3) The original report specified that all industries would sell products in a consumer market. In the present version, the game administrator can specify that an industry produces materials for use in other products.

4) The original report specified that labor relations would have an impact on productivity. This has not been implemented.

5) The original report specified a single representation for relations between entities. In the present version, a number of representations are used.

6) In the original report, the interface between the input program and the simulation program was not specified.

2 GLOBAL GAME DESCRIPTION

2.1 The Infogame world

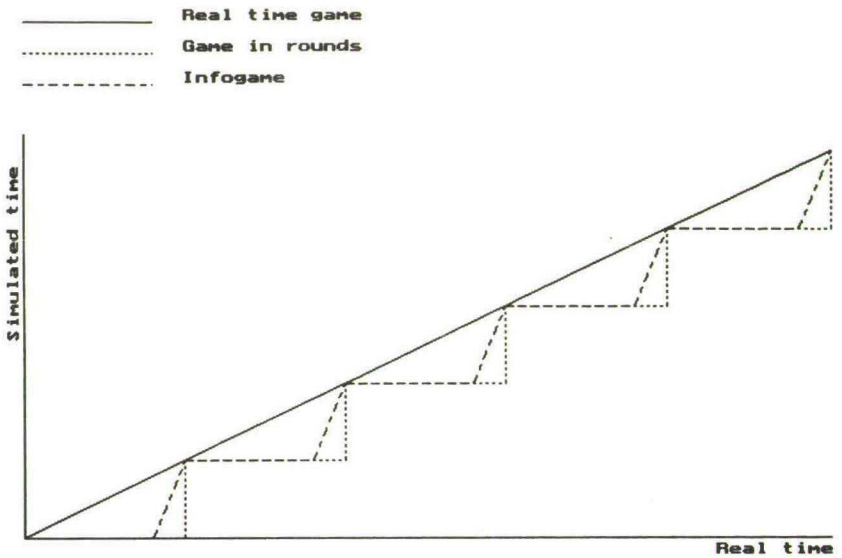
Infogame simulates manufacturing companies in up to five consumer markets. It combines marketing management and production or operations management, a subject that has attracted renewed interest from business schools [4]. From the main problem areas in production management, explicitly listed in [26], but also found in similar texts [4,20]: capacity, plant location, plant layout, capital budgeting, aggregate planning, inventory, scheduling, and maintenance, only plant location is definitely absent from Infogame, as we deliberately abstracted from spatial dimensions. Scheduling and maintenance are simulated by the game program; the other areas are subjects for player decisions. A characteristic of Infogame, which is absent from functional production games, is the interaction between production and marketing management. For example, short delivery times can be used as a marketing instrument, but attaining short delivery times is a production management problem. Labor management is also important in Infogame; this is in accord with European custom, where the overall size of the labor force in a company is considered a top management decision. In contrast to strategic business games, the financial model is simple; expenses must be financed from the initial endowment, income and bank loans. Share transactions, intercompany loans, joint ventures, mergers, sales of assets, and similar topics that dominate the financial pages, are not implemented. However, lack of finance will have dire results for a company, such as a total layoff. Apart from competition in the labor and consumer markets, there is no interaction between players. For example, a company that produces materials cannot sell those to other companies. The reason for this restriction is that contacts with other players would take too much time and energy.

In contrast to many other games, all companies in Infogame start from scratch. This forces players to take decisions such as choice of market and technology, which are intimately connected with operational decisions made in a later stage. For example, a player who chooses to operate in a market with production to order should know he will have to control delivery times. In a game which starts with going concerns, players may survive by just ensuring smooth operations. As it is only fair that all companies start in the same markets with the same technologies, much of the diversity of the Infogame environment would be lost.

2.2 Time scale

Any computer-based game is characterized by the relation between *real time* (the time on the player's watch) and *simulated time* (the time in the world simulated by the computer). Real time is determined by the complexity of the game. Not counting gamaholics, real time for a game ranges from some minutes for simple arcade games to some days for complex management games. Simulated time ranges from thousands of years in historical adventure games to seconds in a computer job-scheduling game. In nearly all games, the player can enter a decision after every simulation of an event or set of events; in real-time games the decision must be entered immediately, because the state of the simulated world is continually changing; in games played in rounds simulation is suspendend during the "thinking" or "decision" period. Infogame is built differently. After the players have entered their decisions, events are simulated on a continuous time-scale. Players may follow events as a motion picture during the actual simulation, but they can no longer influence the course of the current round. In effect, the motion picture interface is not implemented, but at the end of each round, a report is provided listing events in chronological order. The ordering of events is important: a consumer who enters at 12:00 will find a product out of stock if the first batch is brought in at 12:02. The relation between simulated time and real time for each type of game is shown in Fig. 1.

Fig 1: Time Scale



The decision to use event-driven simulation within a single Infogame round was made for the following reasons: First, the detailed reports supplied to players are interpreted more naturally when events can be timed like real world events. This is important, because instruction manuals should describe only in which respects the simulated world differs from the real world. Second, it is difficult or impossible to analytically compute the results of interactions among different types of events. This is shown by the limited results of analytic queuing theory, which is applicable only to a few simple models. Moreover, mathematical models tend to study equilibrium states, whereas we need data on states after a specified number of events [12]. In general, analytical methods such as queuing theory are used as an alternative to simulation, rather than as a technique within simulation modelling.

An example

A trader sells a single non-perishable product. Customer demand is a Poisson process. Every night the stock is replenished to the reorder level. Total sales and average stock during a year should be computed for various reorder levels by simulation. From simulation literature, it is well known that this problem can be solved by two equivalent methods, either by simulating all sales separately, using the negative exponential distribution to draw the time of the next sale, or by drawing the total number of sales from the corresponding Poisson distribution for a day. The two methods are represented by Fig. 2A and Fig. 2B, respectively. Now the specification is changed slightly: we assume that supply is also a Poisson process. The first method can be adapted to the new specification without any difficulty, as shown in Fig. 3. The second method is not applicable without considerable effort, because, even for this simple problem, an analytical distribution for the cumulative results is not known to exist [24].

Fig 2A: Exponential distribution

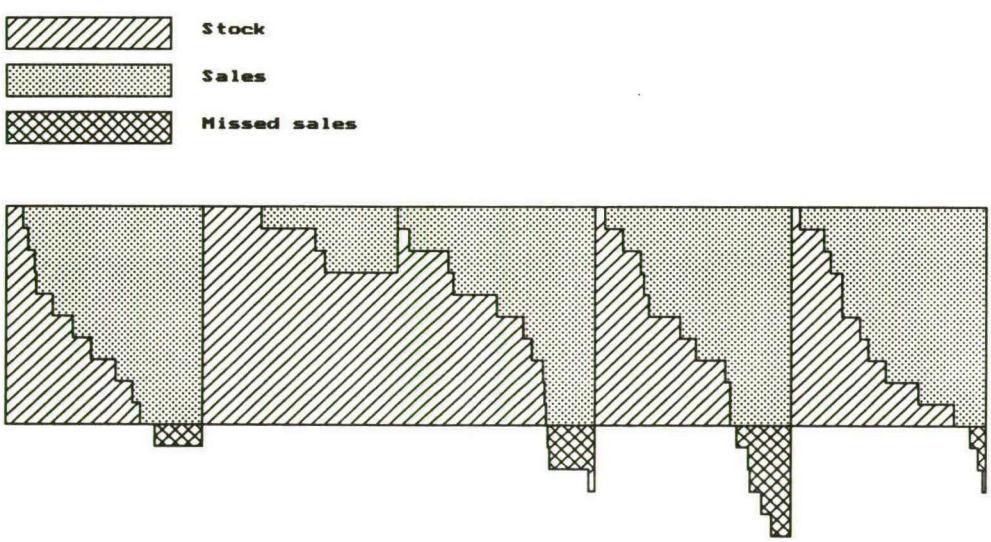


Fig 2B: Poisson distribution

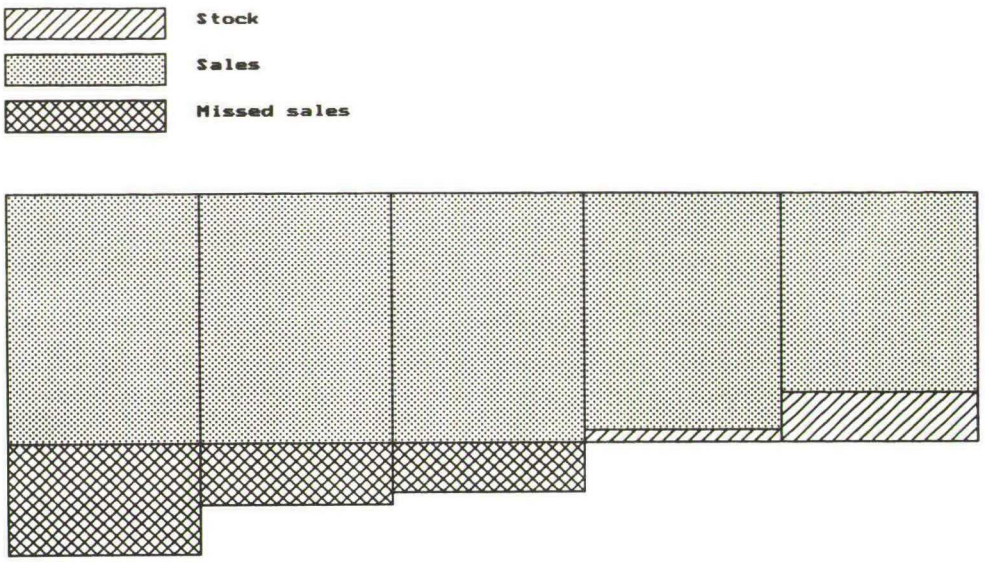
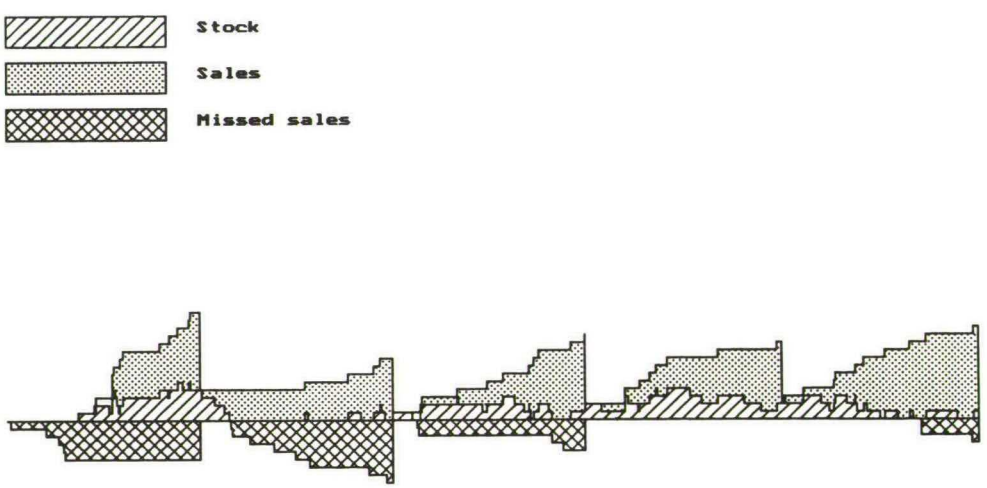


Fig 3: Buying and selling



This example shows that to find analytical distributions for all interactions in Infogame, which usually involve three or more independent processes, would call for a major effort in operations research, which is not our main objective. Moreover, from a modelling and programming point of view, event-driven simulation is an established technique. The programming technique used to maintain the event list is discussed in section 4.

2.3 Instruction by rules

In contrast to a direct instruction, a rule must be followed at a later moment, under circumstances not known when it was defined. This makes rule definition difficult, which is well known to rulemakers such as legislators, programmers, composers, and authors of manuals. Rules are, however, indispensable if decision makers want to control events when they are not on the scene. Accordingly, rules are pivotal to the design of Infogame. There are two types of rules in the game:

a) Fixed rules, which are known to the player, but cannot be changed by him. An example is: "production cannot start unless sufficient materials are available". The player must know such rules because they influence the results of his decisions.

b) Rules with player defined parameters, for example: "an order for a player defined quantity of a material is placed when the stock falls below the player defined reorder level". Production, material buying and reporting are controlled by such rules. Accordingly, many player decisions in Infogame define rule parameters.

For players not used to programming in any form, control by rule is hard to understand; this may even be considered a decisive argument agains Infogame-type games. The original proposals called for an advanced version [5], where rules would be formulated in a programming language or expert system shell language. It is clear that the audience for this type of game is still more restricted.

2.4 Selective reporting

Infogame provides no summary reports, but only raw data on events. For decision making, the player will have to make or buy an information system, because the uncondensed report is unsuitable for that purpose. Reporting is also selective, which implies that output is produced only if specified by a Boolean function t. The output procedure can be described as:

$$\underline{\text{if}}\ t_P\ \underline{\text{then}}\ \text{write}(O_P)$$

t_P Boolean function
O_P Output function
P Type of output

Calls to the output procedure are controlled by explicit calls to the output procedure "possiblereport" in the simulation program:

```
possiblereport(firm,reportcode,descr1,descr2,quantity,price);
```

firm:	Number of reporting firm.
reportcode:	Type of report (e.g. sale, start of production).
descr1, descr2	Strings describing event (e.g. product name).
quantity, price:	Real numbers describing attributes of the event (usually quantity and price).

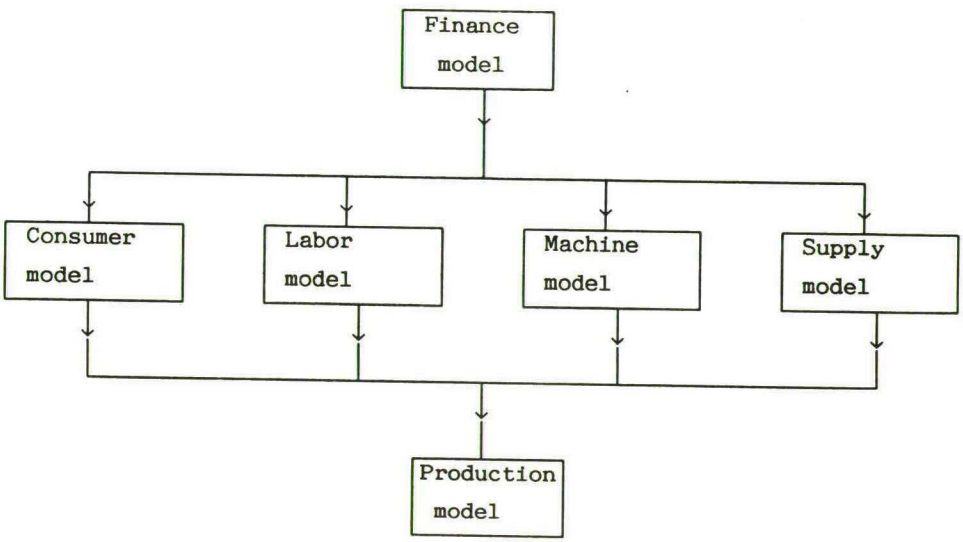
The number and type of the output parameters is fixed, both because in Pascal the number and type of parameters is fixed for a procedure, and because fixed format output is easier to handle, especially in database packages. The meaning of a field depends on the reportcode; for example, in a sales record, "amount" indicates the number of products sold, in a customer payment record, "amount" is the amount paid.

3 THE GAME MODEL IN DETAIL

3.1 An overview

The infogame model is designed as a set of submodels with limited interaction, as pictured in Fig. 4:

Fig. 4: Submodels and interaction



Each submodel describes a sequential process. Processes interact only with the production process and the finance process. There are two types of interaction between processes: processes may consult and update common state variables (for example, the stock level of finished products is consulted and updated by the consumer model and the production model), or a process may cause events in another process (for example, repair of a machine is an event in the machine model which may trigger restart of production in the production model). Implementation details are discussed in section 4.

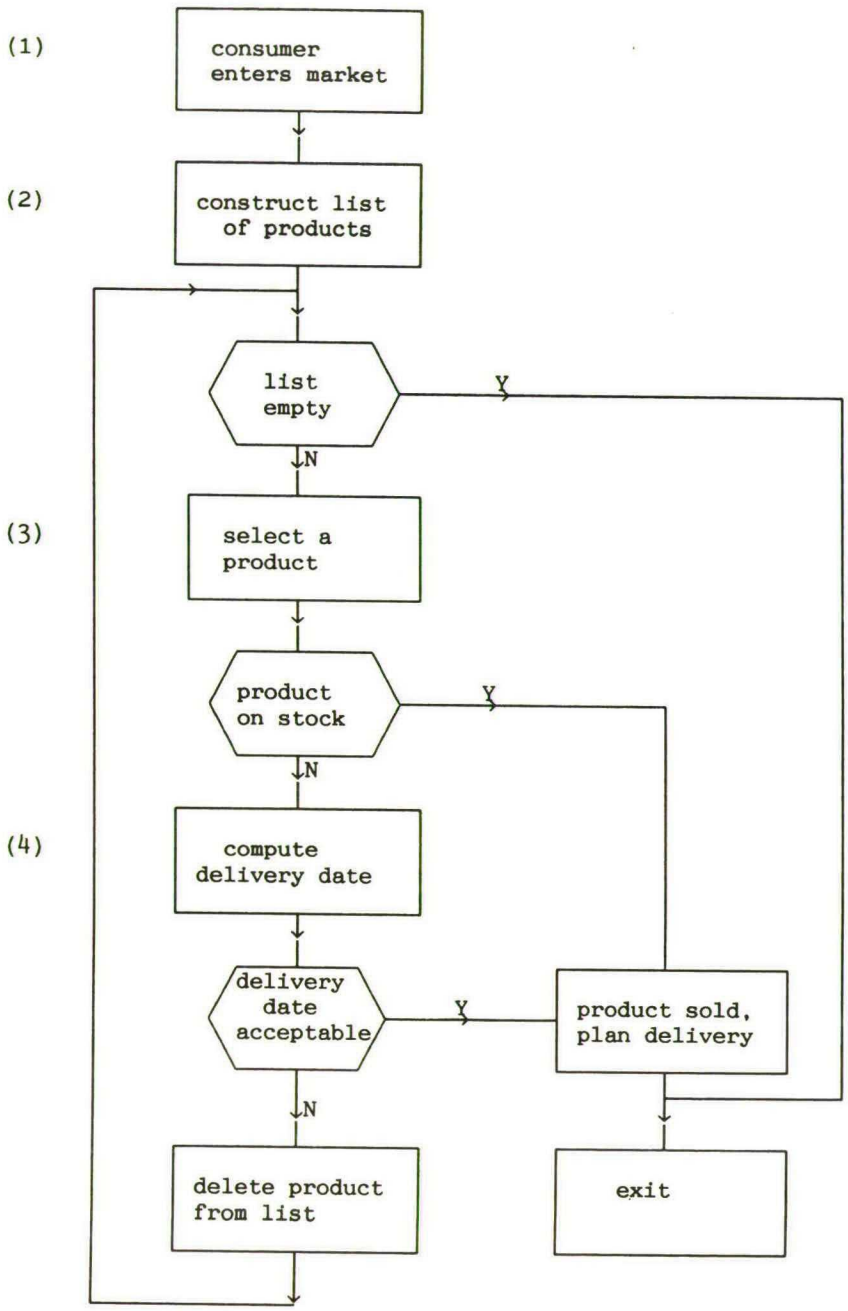
3.2 Consumer model

The consumer model in Infogame simulates individual consumers exercising demand for the products of the firms in the game. The model describes a self-sustaining process: consumers arrive irrespective of the existence of any products. This model has been chosen for several reasons. First, a large number of individual sales has to be simulated anyhow, and simulation of individual consumers is easier than computing an aggregate distribution first and drawing individual events from it afterwards. Second, as explained in section 2.1, an aggregate distribution function may well be difficult to compute analytically; so it may still be necessary to simulate individual consumers to compute the aggregate distribution function. In fact the market model is derived from the market model in MAGEUR [7], which divides the market in 100 segments to simulate market niches. Third, demand may interact with production in a way that cannot be accounted for by an aggregate function (e.g., demand for some products may not be filled because of stockouts). Fourth, users not versed in microeconomic theory may better understand rules describing individual consumer behavior (e.g. consumers choose the best product they can afford), than functions describing global market reactions such as elasticity. However, in a model simulating individual consumers, players cannot directly use the results of microeconomic theory or empirical research based on it. This may be a serious drawback in an environment where practitioners regularly use such results.

From marketing mix instruments listed by textbooks [16], only product quality, price and advertising are implemented. Distribution is left out because we feared inclusion of a real or fictional geography would divert attention from the main features of the game. Promotional efforts other than advertising (e.g. sales-force activities) are included in advertising.

Fig. 5 gives a view of the consumer model.

Fig. 5: Overview of the model:



Numbers in parentheses in Fig. 5 pertain to the following details:

1) Arrival on the market is governed by a Poisson process with mean interarrival time defined by the game administrator. A fairly large number of individual consumers (50-100) is simulated. Consumers are numbered sequentially; lower numbers indicate consumers with less purchasing power. Each consumer exercises demand for the products of a single industry; there is no substitution between industries. Demand is equal to the average of three draws from a negative exponential distribution, with mean defined separately for each industry by the game administrator. The maximum price a consumer is willing to pay and the maximum quality he takes into account are calculated from a Pareto distribution with parameters set by the game administrator:

$$mp_{ij} = b_j \frac{n_j^{c_j}}{n+1-i} \qquad mq_{ij} = (\ln \frac{n}{n+1-i}) c_j$$

mp_{ij} Maximum price for consumer i in industry j (see Fig. 6A).
mq_{ij} Maximum quality for consumer i in industry j (see Fig. 6B).
n_j Total number of consumers in industry j.
b_j Base price (price every consumer can pay) in industry j.
c_j Constant denoting appreciation of high-quality products in industry j, with appropriate values between 0.5 and 1.

The net price to the consumer is influenced by the consumer credit, which is discussed in detail in section 3.7.4. It is defined by:

$$p_n = p_q \left(1 - \frac{dc}{n} \right)$$

p_s Net price to the consumer
p_q Quoted price
d Discountfactor, determining importance of consumer credit
n Number of days per period (quarter)
c Number of days of consumer credit

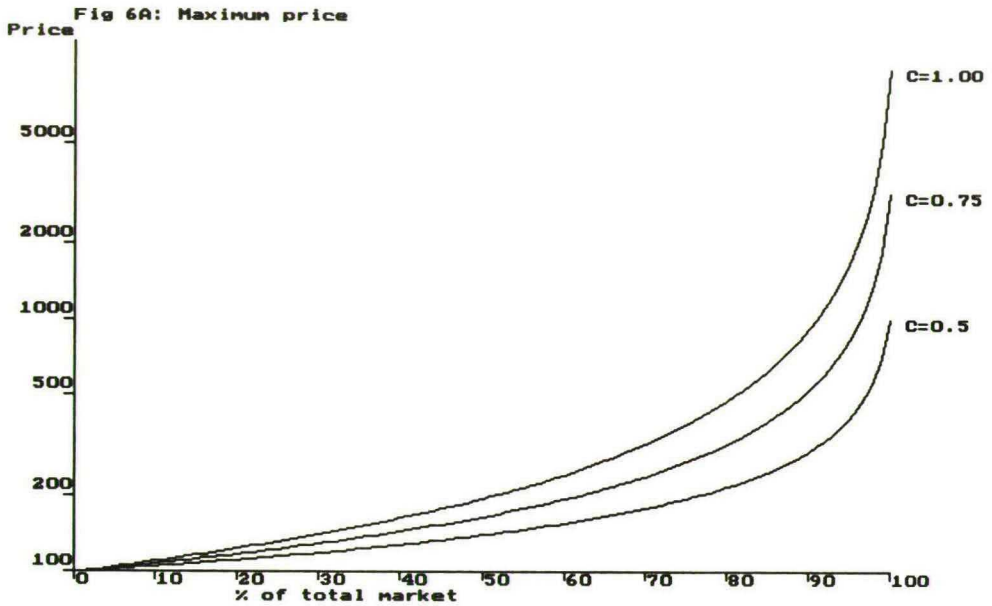


Fig 6A: Maximum price
Price
5000
2000
1000
500
200
100
C=1.00
C=0.75
C=0.5
0
10
20
30
40
50
60
70
80
90
100
% of total market

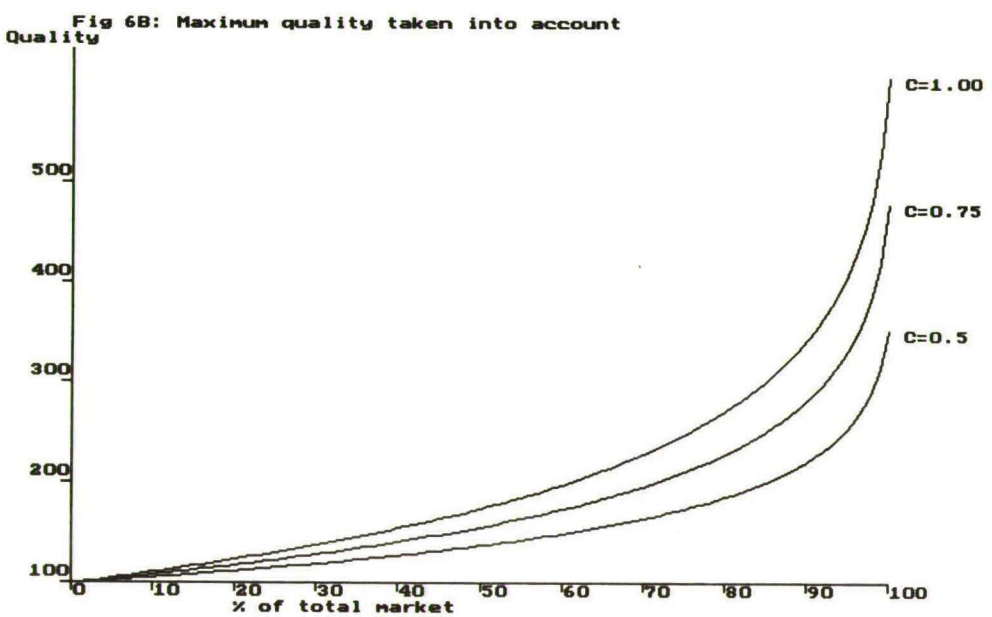
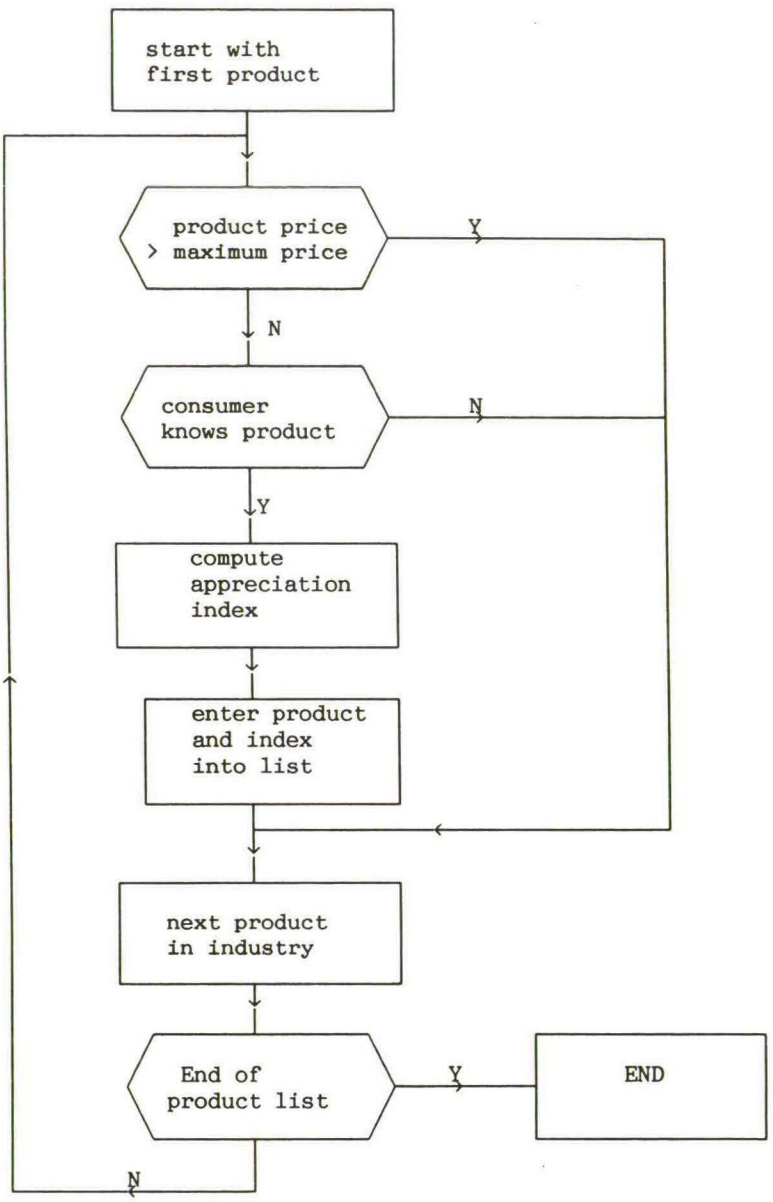


Fig 6B: Maximum quality taken into account
Quality
500
400
300
200
100
C=1.00
C=0.75
C=0.5
0
10
20
30
40
50
60
70
80
90
100
% of total market

2) The list of products is constructed by executing the instructions from Fig. 7 for all products in the industry.

Fig. 7: Construct list of products:



A consumer knows a product from advertising or from experience with the product. A consumer has experience with a product if she or any of her neighbours bought this product the last time. The probability that a consumer is reached by advertising is:

$$p = 1 - (1 - \frac{1}{c})^{E/A}$$

p Probability that a consumers is reached by advertising.

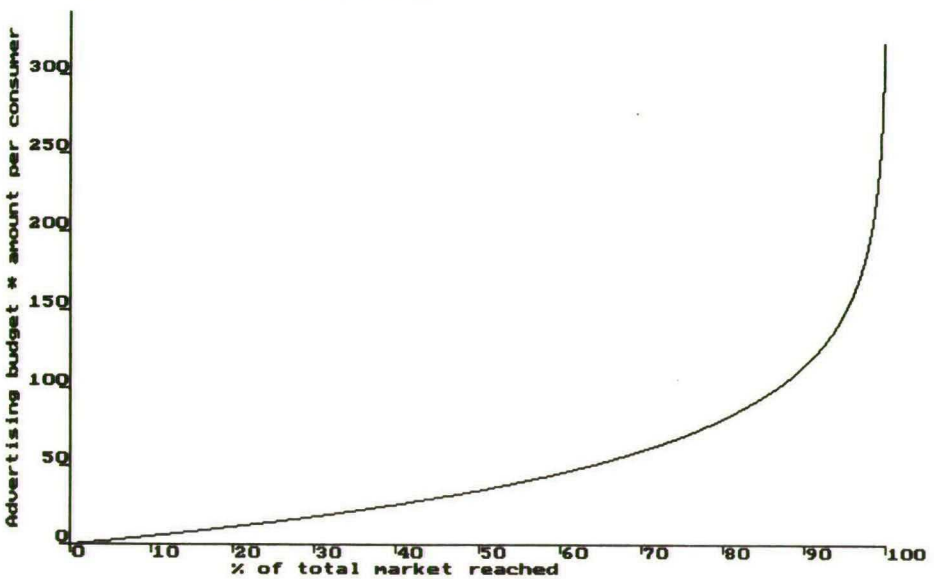
c Number of consumers willing to pay the price of the product.

E Total advertising budget.

A Advertising budget needed to attract one consumer, an industry-specific constant defined by the game administrator.

Graphically, the proportion reached is shown in Fig. 8

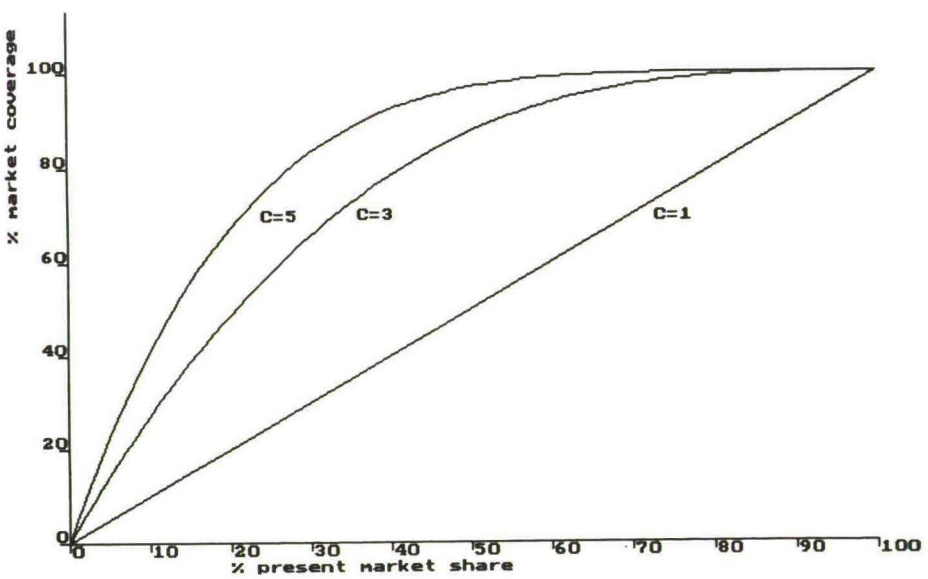
Fig 8: Result of advertising



It is assumed that advertising is always directed at the appropriate target group only, i.e. consumers who are willing to pay the price of the product. This implies that advertising for low-priced products is directed at the population at large, whereas advertising for high-priced products is directed at a limited group. This is not unrealistic, as in the real world a choice must be made between mass media, such as television, and selective media, such as specialized journals, trade shows and direct mailing. In Infogame, this choice is not made explicit, because selection of advertising media is not considered a task for top management.

The number of neighbours in each industry is determined by the game administrator. They are located above and beneath a consumer on the income scale. This implies that advertising is needed to reach customers in an income class not yet served by the product. The number of neighbours and the present market share determine the part of the market that is reached without advertising, as shown in Fig. 9.

Fig 9: Market coverage



A consumer will consider all products priced at or below his maximum price for inclusion in the list of possible products. Products not on stock may be included, because some consumers will accept waiting for some products. Moreover, the producer has to be contacted to establish whether a product is on stock, or to quote a delivery date. Although some consumers are ready to pay a considerable premium for a superior product, there is no minimum quality, as most consumers will prefer low-quality products to nothing.

For each product, the appreciation index is computed by the formula:

$$A = \frac{q^n}{p^m}$$

A Appreciation index.

q Corrected quality, i.e. the maximum of the actual quality of the product and the maximum quality taken into account by this consumer.

p Price.

m,n Industry-specific constants, defined by the game administrator; typical values are n=6, m=3.

It is an essential characteristic of the market model that the appreciation index is computed separately for each consumer. Consequently, there is normally no single product that is considered best by all consumers

The resulting list may contain all products in the industry; condensing it to a short list before proceeding with the actual selection has no significant advantages over direct selection.

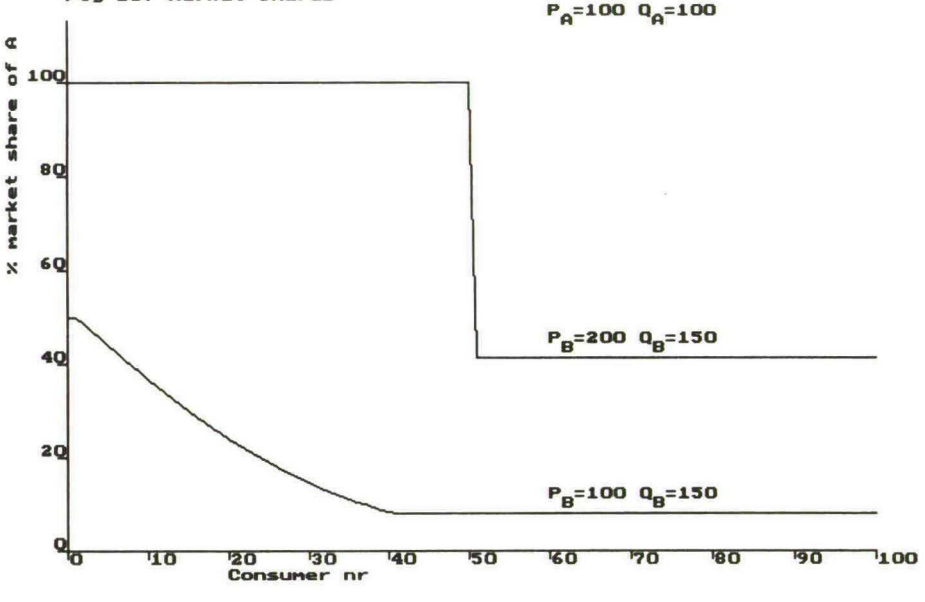
3) A product is selected such that the probability that product i will be chosen from a list containing n products is:

$$P_i = \frac{A_i}{\sum_{j=1}^{n} A_j}$$

A_i Appreciation index of product i.

Selection is done by linear search because complexity is linear anyway as it is dominated by the computation of appreciation indices. Fig. 10 gives the probability that consumers will choose product A from products A and B.

Fig 10: Market Shares



4) The quoted delivery date is supplied by the player, who has to choose between the risk of losing customers by quoting long lead times and the risk of missing quoted delivery dates. For each consumer, a maximum

delivery time is drawn from an uniform distribution with industry-specific bounds defined by the game administrator. In industries that produce for stock only, upper and lower limit have zero values. The quoted delivery date is, however, not taken at face value by consumers, but it is divided by a reliability factor which is computed as follows:

$R_0 = 1$

$R_i = \max(0.01, m.R_{i-1})$ If a delivery is not in time

$R_i = m.R_{i-1} + 1-m$ If a delivery is in time

R_i Reliability factor after delivery i

m Memoryfactor, an industry-specific variable between 0 and 1, defined by the game administrator.

Because of its influence on the reliability factor, failure to deliver in time can make a product inacceptable to all consumers in an industry. A simulation study has shown that this occurs regularly if the delivery time is set so low that demand is higher than production capacity, but so high that even a slight fall in reliability makes the delivery time unacceptable. This state can only be corrected by reducing the quoted delivery time, perhaps to zero.

When a consumer chooses a product, the maximum delivery time is a hard criterion: products with longer delivery times are not considered, but otherwise consumers do not prefer products with a shorter delivery time. When a product is ordered, consumers accept a delivery time equal to their maximum delivery time multiplied by an industry-specific slack factor defined by the game administrator. When production is to order, a product that is ready for delivery past the acceptable delivery date cannot be sold on the consumer market, so it is dumped without any extra cost or income. However, production will not be started after the acceptable delivery date. Scheduling of deliveries is discussed in section 3.6.

3.3 Labor model

Labor relations are explicitly implemented in Infogame. The player must decide on the hiring and firing of employees. In accordance with European labor policy, employees cannot always be dismissed immediately. Each player sets a term defining the time (in salary periods) an employee will stay with the company after being given notice. The same term will apply to employees: they have to give notice beforehand if they want to take another job. There is only one type of employee. Productivity of new employees is lower than productivity of existing employees; otherwise, productivity of all employees is the same.

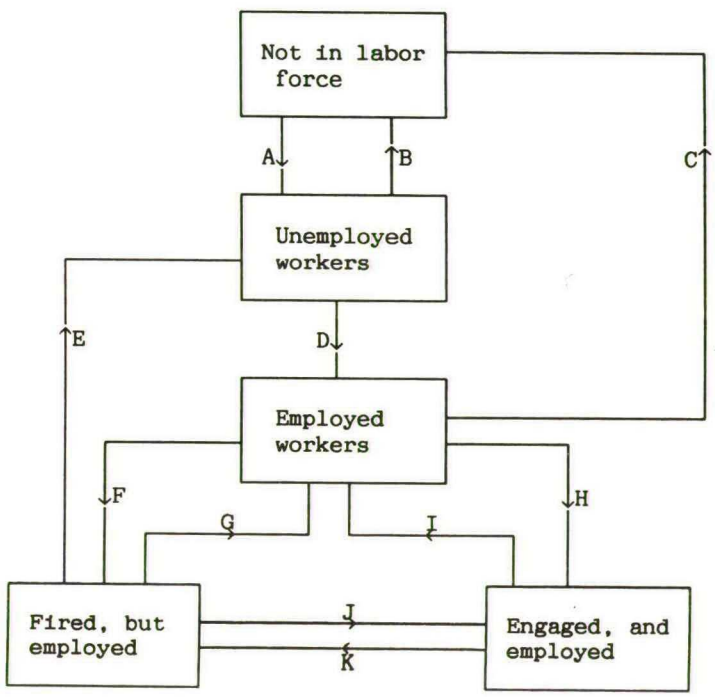
The labor model differs from other submodels as it is time-driven rather than event-driven. Workers start, change, and terminate jobs only at the start of twenty-day salary periods (months). This is done because of the computational complexity of the labor market model, where a large part of all workers consider jobs on offer. It is also in accord with Dutch practice, were workers are hired by the month. Strikes are not implemented in Infogame, though they may have a large impact on company affairs, and are a regular feature in European management games [23]. First, the duration of a strike is normally much shorter than a simulated game period, so the outcome of a strike should be decided during a round, which is only possible in a real-time game. Second, whereas it is easy to model the stakes of an employer in strike negotiations, it is very difficult to model the motivations of the opposing labor officials. Consequently, human players acting as labor officials would have to rely on role-playing.

The model is described in two stages. First, states and relations between states are described in a global model. Next, the operational model, that implements the global model by sequential simulation, is described in detail. Of course, sequential simulation is not the only way to implement the global model, but in the Infogame context it is more appropriate than parallel simulation or solution by solving a system of simultaneous equations.

Global model description

In our model, which is based on [13], persons outside the labor force enter it as unemployed workers. All unemployed workers and a part of all employed workers are attracted to new job opportunities. The special states "fired but still working" and "hired but still working elsewhere" are introduced because both employers and employees may have to give notice one or more months before a job is terminated. The model also recognizes the state "not in labor force", but it does not count the number of persons outside the labor force; the number of persons entering or leaving the labor force is determined only by the number of employed and unemployed workers, not by the number of persons outside the labor market. Fig. 11 gives the model.

Fig. 11: Labor Market model.



The following assumptions are made:

a) Persons joining the labor force start unemployed, but they may be employed immediately, because the model computes the addition to the labor force before the hiring of new workers. At the beginning of the first period the labor force contains only unemployed persons. The size of the initial labor force is determined by the game administrator; a typical value is 500.

b) Unemployed workers join an employer as soon as they are hired. Employed workers join a new employer only after finishing their term with their present employer, even if they are fired.

c) Employers can fire a worker only after a fixed term.

d) Both employed and unemployed persons can leave the labor force at any moment, but no one will do this while changing jobs. Employed workers do not go into voluntary unemployment.

Transitions in the model (see the letters in Fig. 11) are:

A,B) Unemployed persons joining or leaving the labor force.

The number of persons joining or leaving the labor force is determined by the formula:

$$J = a(T - U + cW)$$

J Number of persons joining the labor force. If $J<0$ then $-J$ is the number of persons leaving the labor force.

a Adjustment factor. A parameter defined by the game administrator determining the reaction of supply to demand. If a=1, supply reacts immediately to demand, so the market is totally open, as employers can get all the employees they need. A more typical value is a=0.2.

T Total number of employees wanted. This is equal to the sum of the number of employees employers want to hire in the current quarter. This total is not corrected for employees fired by other employers.

U Number of unemployed workers.

c Correction constant, defining an automatic increase of the labor force. A typical value is c=0.04.

W Total number of (employed or unemployed) persons currently in the labor force.

C) Working persons leaving the labor force.

The probabilty that an employee leaves the labor force is a constant determined by the game administrator. A typical value is 0.03.

D,G,H,J) New workers hired.

An employer can hire unemployed persons, persons who are fired by an employer (including himself) and persons actually employed by other firms. The employer's success is determined by a number of factors that will be detailed in the operational model. The player determines the number of workers he wants to hire. This number is interpreted as a net number; it is increased with the number of employees leaving in the same period. This approach is chosen to prevent a mere exchange of personnel between firms instead of an increase in employment; a disadvantage is that it is not possible to partially replace the employees that are expected to leave. Employment of unemployed workers or recall of workers previously fired by the firm itself takes place immediately.

F,K) Workers fired or not hired.

We assume that an employer having too many workers first stops hiring. Workers already hired, but still employed by another firm, are considered fired by their former employer. The number of workers fired is determined by the player.

E) Persons losing employment.

Workers made redundant and not hired during the redundancy term become unemployed.

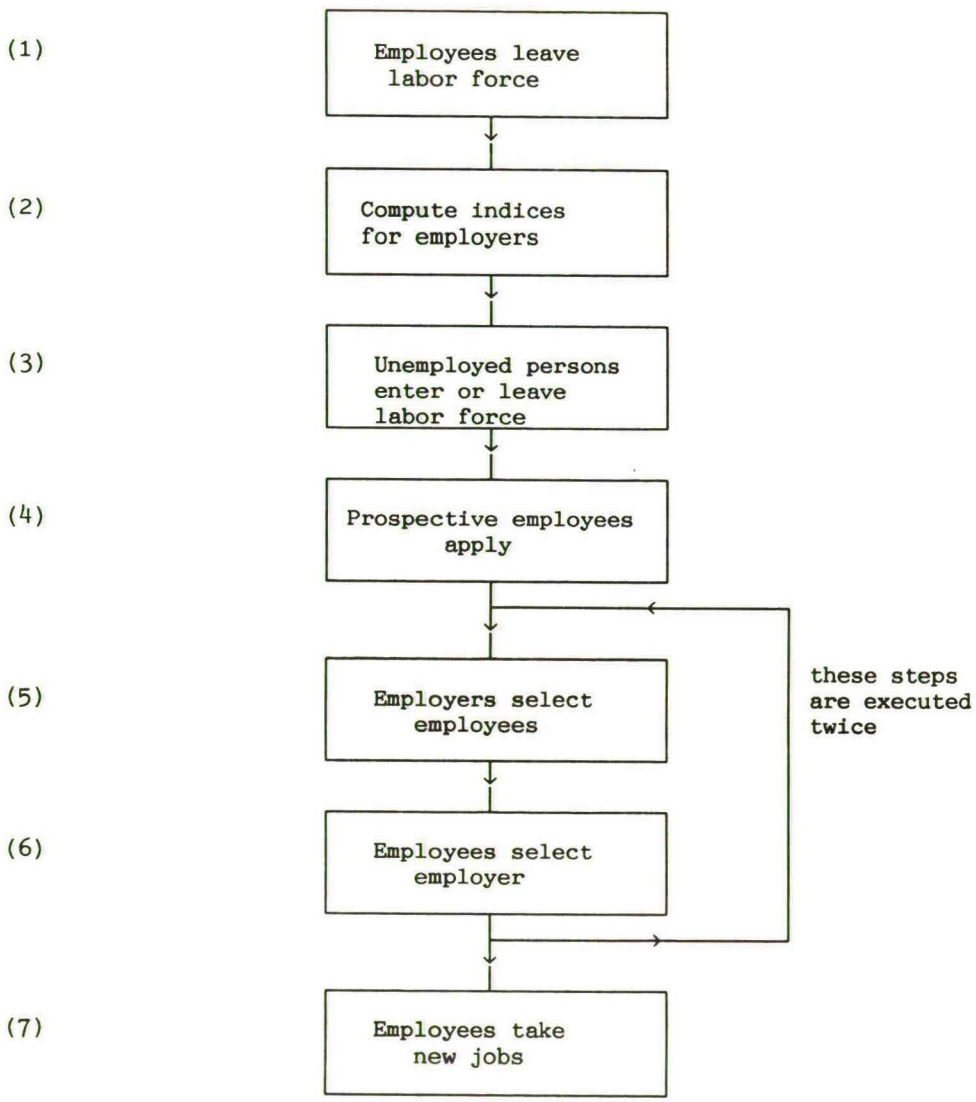
I) Persons joining a firm.

Workers hired away from another firm start with their new employers after serving their term with their present employer.

Operational model description

The labor market is simulated in seven steps as in Fig. 12

Fig. 12: Operational model of the labor market



Steps (1) and (3) are implemented as loops that range over all persons. Step (2) is executed as a single loop ranging over all firms. Steps (4), (5), and (6) are implemented as double loops. In steps (4) and (6) the outer loop ranges over all persons and the inner loop ranges over all firms. In step (5), the outer loop ranges over firms and the inner loop over persons. Consequently, the loops cannot be merged without changing the model. Step (7) is implemented by executing events initiated by the preceding steps. Computations performed in steps (1) through (7) are:

1) This step models transition C) from the model in Fig. 11.

2) In this step the following computations are performed:

a) The data determining the expansion or contraction of the labor force, which is actually effected in step 3, is collected. This part of the model is described for transitions A) and B) of Fig. 11.

b) For each firm, the probability that the attention of an (employed or unemployed) person is drawn to the vacancies is computed by the formula:

$$P = 1 - 0.99^{T/C}$$

P Probability that a person's attention is drawn.
T Advertising budget for recruitment.
C Cost of reaching 1% of potential workers.

This formula is analogous to the formula used in the consumer product advertising model. It guarantees diminishing returns of recruitment expenses and a limit of 100 % to exposure. Workers who have been fired but are still working for the company are attracted to new vacancies without advertisement. The full personnel advertising budget is spent at the start of a period, but the resulting attention remains the same during the whole of the period.

c) For each firm, a satisfaction index is computed by the formula:

$$S_i = mS_{i-1} + (1-m)\ \max\left(0, \frac{w}{\bar{w}} - \frac{f}{p+f} + d\sqrt{D}\ \right)$$

S_i Satisfaction index in period i.

m Memory factor used for exponential smoothing; typical value is 0.8.

w Salary level of this firm.

$\bar{w}$ Average salary level of all firms.

f Number of workers actually fired in this period, i.e. the number of employees who have been fired, and have not found a new job during their redundancy term.

p Number of employees in this firm.

d Delayfactor, indicating the importance of stable employment for employees. A typical value is 0.1.

D Length of notice term in salary periods.

The satisfaction index is used in step (4) and (6) to compute the probability that a worker will switch jobs.

4) A prospective worker whose attention is caught by the employer, will always apply for the job if unemployed or recalled by his present employer. A recall pertains to a worker who has been fired but is still employed, which differs from American usage where workers are fired immediately and may be recalled later from unemployment. For a worker employed by another employer, the probability that he will apply is equal to:

$$\max\left\{0,\ R\left(1 - \frac{s_c}{s_p}\right)\right\}$$

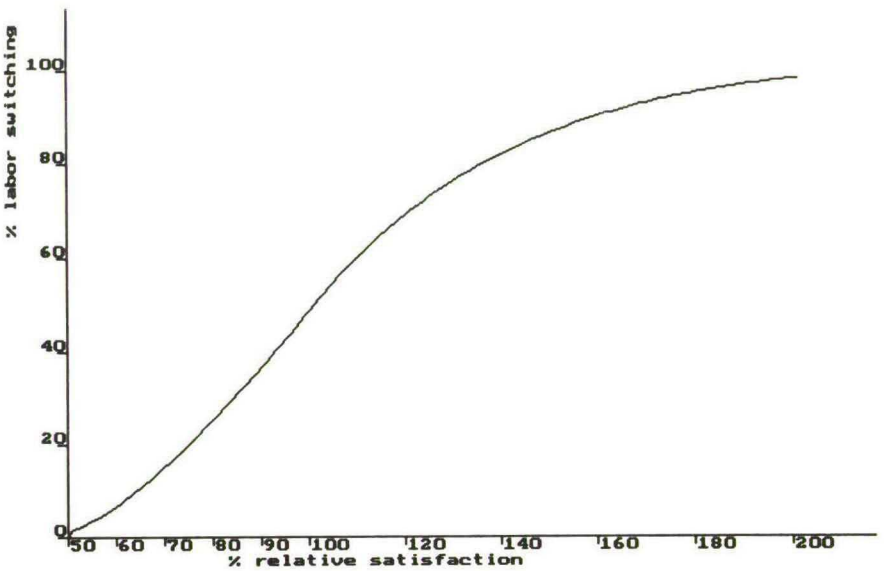
R Response index, a constant ≤ 1.

s_c Subjective satisfaction index for the current employer

s_p Subjective satisfaction index for the prospective employer

If the satisfaction index computed in step 3c) would be used as the subjective satisfaction index, employees would only apply for jobs with employers with a higher satisfaction index. To model the phenomenon that, at the same time, some workers switch from A to B, and others switch from B to A, subjective satisfaction indices s_c and s_p are drawn from uniform distributions $[0.6,1.4)S_c$ and $[0.6,1.4)S_p$ respectively. The probability of a switch between firms is shown in Fig. 13

Fig 13: Labor market switch



5) An employer who has more vacancies than applications, offers a job to every applicant; otherwise one applicant is selected for every vacancy. This is a cautious policy, as it is not certain that all job offers will be accepted. Job offers are made in two stages. First, jobs are offered to all employees who have been fired and are not engaged elsewhere; after that, offers are made to outside applicants.

6) A person who has received a single job offer will always accept it. The probability that a person who has received job offers from firms 1..n will choose the offer from firm i is equal to:

$$P_i = \frac{S_i}{\sum_{j=1}^{n} S_j}$$

P_i Probability the offer from firm i is accepted.
S_i Satisfaction index of firm i

Because the choice process itself contains a random factor, the objective satisfaction index from step (3) is used instead of the subjective satisfaction index used in step (4). An employee does not always prefer a recall over a move to another job. Contrary to American practice, we do not assume that unemployed workers have a preference for their last employer.

When employers compete for a limited number of workers, less workers are hired than needed; this is partially corrected by executing steps (5) and (6) twice during a salary period (month) and by filling remaining vacancies in subsequent salary periods. Nevertheless, most workers will be hired at the start of the period (quarter).

7) Acceptance of a job offer triggers a state change. Unemployed persons take their new job without delay and recalls are also effected immediately. Employees with other firms take their new jobs only after the necessary term. For simplicity, this also applies to employees who have been fired: If a person is fired in month 4 with a three-month term, and accepts a job with another firm in month 6, he starts in his new job in month 7.

Labor as a resource

The unit of labor in Infogame is the fully productive worker. The number of fully productive workers in a firm is equal to the product of the actual number of workers and the productivity index (a number between 0 and 1). The number of employees is computed straightforward from the number in the preceding salary period (month) by adding arrivals and subtracting departures. The productivity index is computed each month with the formula:

$$P_i = \frac{(w_{i-1} - d_i)\left[P_{i-1} + \ell(1-P_{i-1})\right] + n_i P_1}{w_{i-1} - d_i + n_i}$$

P_i Productivity index in month i.

P_1 Productivity index of new workers, defined by the game adnministrator; a typical value is 0.5.

w_i Number of workers in month i.

d_i Number of workers departing in month i ($d_i \leq w_{i-1}$).

n_i Number of new workers arriving in month i.

ℓ Learning factor, defined by the game administrator; a typical value is 0.3.

Because $w_0 = 0$ and $d_1 = 0$, P_1 defines the productivity of new workers and the productivity in the first salary period. It is easily seen that the value of P_i ranges between P_1 and 1.

If the number of employees increases, new employees can only be employed on new jobs; if machines are idle and materials are available, arrivals may trigger the start of a job. Conversely, departing workers are assumed to finish their jobs first. The two assumptions were introduced to simplify the model; with average job times, they have no significant effect on results. Similarly, the productivity index is only used for new jobs; current jobs continue with the assigned workers and are finished at the scheduled times.

3.4 Machine model

In Infogame, machines are resources like workers, but the machine model is implemented differently, both because machines are different from people in the real world, and because the level of complexity of the submodels should not differ too much. The production unit in Infogame is a factory employing a small number of expensive machines and a fairly large number of operators. The differences between workers and machines are:

1) All workers are equal, whereas there are many types of machines. In the real world, workers are more flexible than machines in many industries, but in offices, specialist workers use standard equipment. The main reason for not introducing different types of workers is that this would make the labor submodel too complex.

2) Machines are bought and can be used until they are scrapped. The only additional costs are repair costs. Employees are hired and must be paid monthly as long as they are employed. A firm can cut costs by firing employees, but reduction of machine capacity by scrapping machines brings no revenue; this reflects the facts that, in the real world, the second-hand equipment market is of minor importance, and that it is difficult to model this market.

3) Workers are hired on a competitive labor market; a machine is just ordered and, if it can be paid, is installed after a delivery time drawn from a negative exponential distribution with mean MTTD, the mean time to deliver, defined separately for each machine type by the game administrator.

4) Machines are treated individually, whereas employees are assumed to belong to a general labor pool. This is in line with the fact that the number of machines is much smaller than the number of workers.

5) Failure of machines is explicitly implemented, whereas illness and other factors affecting labor productivity are not.

Any machine will fail after a time drawn from an negative exponential distribution with mean MTBF (Mean Time Between Failures) determined by the game administrator. If the machine is idle at the time of failure, repair starts immediately, otherwise, repair starts when the job using the machine is finished. This simplification can be explained as follows. In the real world, a machine breakdown may occur at any time. A machine that breaks down when employed on an urgent job will be replaced by a machine that is either idle or employed on a less urgent job. So either the least urgent job that actually uses this type of machine will be finished later, or a job that will use it will have to wait longer before starting. In the real world, rescheduling in case of machine failure is the task of a low-level manager; consequently, in Infogame it should be executed automatically. The rescheduling algorithm is fairly complicated in itself. It also supposes preemption, which entails additional complexities. Thus rescheduling in case of machine failure should be implemented only if its use materially influences scheduling results. A simulation study has shown no significant differences in the distribution of total time from the moment a job is placed in the queue till it is finished for the two models, so the simple model (failures do not occur during the use of a machine) was chosen.

Repair time is drawn from a negative exponential distribution with mean MTTR (Mean Time To Repair) defined by the game administrator. Repair cost is linearly related to repair time. When the machine is repaired, it is put back into use if the repair cost can be paid; otherwise it is held back until the repair can be paid. After repair, a machine cannot be used until a new job is started. If machine capacity is a critical factor, repair of a machine may trigger the start of a new job.

A machine can be scrapped at the start of a period, but is will not be scrapped while employed on a job; the reason for this restriction is the same as that for suppressing machine failures during operation.

If a machine is in repair when the order to scrap it is given, it is scrapped when the repair is ready, but the repair cost must still be paid.

Machines are not subject to economic aging. Technical aging is implemented by a decrease of MTBF and an increase of MTTR after each period. Initial values and rates of increase for MTBF and MTTR for each machine type are set by the game administrator. Replacement is a standard problem in Operations Research textbooks. The OR approach can be used by the game administrator to determine whether his data will favor replacement during the normal playing sequence of 10 to 20 rounds.

3.5 Supply model

Infogame distinguishes a number of different types of materials, each denoted by an integer number. Internally, the type of material is equivalent to the quality of a finished product. Every process defines the types of material and the quantities needed to produce a unit of the product. For each type of material, the game administrator defines suppliers and their characteristics. The player defines the reorder level, the order quantity and a nonempty list of suppliers for each type of material. If more than one supplier for a material is appointed, orders are allotted to suppliers in round robin fashion. If a supplier is removed from the list of suppliers, no new orders are given to this supplier, but existing orders are executed normally. A new supplier who is added after the first period will not get the first order, but otherwise his place in the chain is random.

At any moment when the stock of the material, including the amount ordered previously, is below the reorder level, an amount equal to the order quantity is ordered from the indicated supplier. For technical reasons, the order quantity cannot be lower than a minimum defined by the game administrator; otherwise, u multitude of small orders could cause a heap overflow.

Ordering can occur only at the start of a period (when the reorder level may have been increased), after allocation of material to a production job (see 3.6), and when an supplier responds to an order with a refusal to deliver, either because cash delivery has been agreed and the firm cannot pay, or because the credits extended to the firm already surpass the specified maximum. After ordering, the selected supplier will respond after a normally distributed delay, either by delivering the material, or by refusing to deliver. After delivery, materials are stored without handling cost in a general warehouse with unlimited capacity. When needed, materials are assigned to a specific job.

Because production of some products may be stopped for lack of materials, delivery of materials may trigger a production restart. Following a change in technologies applied, some materials may become superfluous. Those materials will be held in stock forever; they cannot be sold or dumped.

Materials can also be produced by the firm itself. In that case, an order is translated into a production order for the specified material, which is executed in the same way as a production order for a consumer product (see 3.6). When the order is finished, the amount produced is added to the stock of materials. If the game administrator specifies that for some materials a choice can be made between in-house production and outside suppliers, players are forced to make a buy-or-make decision. On the other hand, if materials needed in final products can only be made in-house, materials management becomes a prime concern.

3.6 Production model

The basic notion in the Infogame production model is the process type or technology, it defines the number and type of machines, the number of workers and the amount and type of materials needed to produce a specified number of units of a product of a given quality in a given industry.

The relation between products and technologies is many-to-many: a product may be produced by several technologies that deliver a product of the same quality in the same industry, and a technology may be used to produce several, technically identical, products.

Production is started by executing an order from a queue containing all production orders. If production is for stock, a production order is issued if the stock falls below a player-specified level. When production is to order only, a consumer order is directly translated into a production order.

There are no priorities in scheduling. For each order, the latest acceptable delivery date is determined, but that is only used to determine whether the order should be executed. Production orders are reviewed when resources become available (machines are installed or repaired, new workers arrive, materials are delivered, or operators and machines are released when a job has been finished), and when an order is entered into the queue. The selection procedure sequentially examines all orders in the queue until one has been found that can be executed with the available resources (or none has been found). In the operating systems literature [3] this method is known as "complete allocation of resources". It has been proved that it prevents deadlock, but allows "starving" of a process by two or more processes that alternately use one of its necessary resources. In view of the aims of Infogame, this is no defect of the scheduling algorithm: starving is considered a sign of resource shortage, which should be remedied by acquiring the appropriate resources. A side effect of the selection process is that orders are deleted if the latest acceptable delivery date has passed. This applies only to industries with production to order. In industries with production for stock, the latest acceptable delivery date is set to infinity.

An order can be executed if two conditions are met. First, sufficient materials must be available. Mathematically, this is expressed by the condition:

$$\forall\, i \in M:\ q_i \geq \frac{r_i\, B}{C}$$

q_i Stock of material of type i.

r_i Materials of type i needed for production at full capacity.

M Set of types of materials.

B Order size.

C Production capacity for a period (quarter).

Second, production time should be finite. Production time is expressed by the formula:

$$t = \min\left(\frac{B.n}{\min\left(\frac{w}{W}, \frac{m_i}{M_i}\ i \in I, 1\right).C}, x\right)$$

t Net production time for a batch in days.

B Order size.

w Number of fully productive operators.

W Number of operators needed for production at full capacity.

m_i Number of machines of type i in operation.

M_i Number of machines of type i needed for full capacity.

I Set of machine types.

C Production per quarter at full capacity.

n Number of days per quarter.

x Maximum time specified.

In industries with production to order only, production time is considerd finite only if it is not limited by the maximum time allowed.

In industries producing to order, a startup time is always added to production time. In industries producing for stock, no extra startup time is needed if a production job immediately follows another job producing the same product with the same machines.

In all other cases, for example if production is stopped for any time, or if one of the machines needs repair, startup time is added to net production time. Startup times are defined separately for each technology by the game adnministrator. A technology with high startup times demands reliable machines and sufficient stocks of materials.

When a production order has been finished, there are two possibilities. If production is to order only, the first corresponding consumer order will be met. If production is for stock, the stock will be replenished, and it will be used first to fill current backorders. A consumer order will not be filled if the actual date is past the latest acceptable delivery date. Failure to deliver results in lost sales and a lower reliability factor (see 3.2). Morover, if production is to order only, the product cannot be sold elsewhere.

It is clear that this scheduling model is not optimal. First, it does not use foresight in planning. For example, a job may be started even if some critical resource will become available some time after. Second, it does not support priorities; jobs are started without regard to the latest acceptable delivery date or other critical factors. A simple model has been chosen mainly for ease of implementation, but it has the added advantage of forcing players to give the necessary attention to major planning decisions; contrary to real-world practice, errors in planning are not corrected by lower level management. Originally, we planned to incorporate Artificial Intelligence planning and scheduling algorithms [10,19] into an advanced version of Infogame; our present view is that research in this line should be done in real-life environments.

3.7 Finance model

3.7.1 Receipts and expenses

The financial model of Infogame is quite simple. Receipts cause an increase in the cash level of a company. The only events that cause a receipt are payment by consumers, interest payments by the bank and receipt of loans from the bank. When an expense is scheduled, the cash level is checked to determine whether the expense can be made. If enough cash is available, the cash level is decreased with the planned expense, otherwise the consequence depends on the type of event; the events causing expenses are listed in table 1:

Table 1: Consequence of insufficent cash

Event	Consequence
Personnel advertising	No advertising
Consumer advertising	No advertising
Cash materials delivery	Materials are not delivered
Payment for materials	Payment rescheduled, supplier credits may become exhausted
Machine delivery	Machines are not installed
Repair cost	Machine cannot be used, repair rescheduled
Salary payment	All workers fired, payment rescheduled
Loan repayment	Cash level becomes negative
Interest payment	Cash level becomes negative
Accounting cost	Cash level becomes negative

A company without cash or credit effectively stops operating. To facilitate post-mortem investigations, reporting is never halted for financial reasons.

3.7.2 Bank transactions

In Infogame the maximum loan amount is defined separately for each firm by the game administrator before each round. Because maxima can only be defined for existing firms, no firm can get a loan in its first round. The game administrator can change the interest rate for long term loans for each firm separately. The interest rate for short term loans and the default rate for long term loans are defined by the game administrator at the start of play.

The bank provides two types of loans: long-term loans and short-time loans. A long-term loan is granted for a specific number of periods at a fixed rate of interest. When it is granted, cash is supplied to the company; At the end of each period, a proportional part of the loan is repaid. Interest over the remaining part of the loan is paid at the end of each period. Short-term loans are granted in the form of limits to overdrafts. Interest payments are computed daily from the current cash balance, and accumulated to be paid at the end of an interest period. If an overdraft is granted, the cash shown in the report is negative, but expenses are allowed as long as the overdraft does not exceed the agreed limit.

3.7.3 Payment scheduling

When a scheduled payment by the firm is due, a check is made to determine whether it can be made. If payment is not possible, the action given in table 1 is executed, and the payment is rescheduled with the delay defined by the game administrator (a typical value is 20 days). Accordingly, an obligation to pay never disappears until it is paid. The result of a failure to pay depends on the type of payment. If salaries are not paid, all employees are fires automatically, and it will cost the firm dearly to attract and train replacements; if a supplier is not paid, the only result is that no more can be bought from this supplier. As a player has no control over his expenses, he should always expect the worst possible consequence of a cash shortage.

3.7.4 Consumer credit

After a delivery, a consumer pays promptly if not given credit. If a credit is extended, the consumer will pay after a delay drawn from a negative exponential distribution with mean separately determined for each industry by the game administrator. If this delay is longer than the maximum delay specified for the given industry, the consumer does not pay at all. The probability that a consumer will fail to pay is:

$$P = e^{-M/A}$$

P Probability a consumer will not pay.
M Maximum payment delay.
A Average payment delay.

For example, the probability that a consumer will not pay is less than 1% for M = 5A.

If the delay is shorter than the credit term defined by the firm, the consumer will pay at the date specified by the firm. Accordingly, the average delay to payment is defined by:

$$D = C + A.e^{-C/A}$$

D Average delay to payment.
C Duration of credit extended by firm
A Mean delay of consumer.

3.8 Operational accounting

As stated before, accounting is the player's responsibility. In a game that is designed for the development of decision support systems but assumes the existence of reliable operational information systems, the course of events in the simulated world should not depend on a player defined accounting system. In fact, it is possible to run Infogame

without getting any information beyond the daily cash statement. Ideally, any datum on which operations are based should correspond to a state in the simulated world that can be obserrved directly in the real world, such as the stock level or the number of employees of a firm.

Variables that record those states can be consulted by the Infogame simulation program; they can also be used for inventory reports to the player. Some facts, however, such as the existence of a consumer order, are observable only by consulting some file. In Infogame, such files are implemented as lists, that can be consulted by the simulation program, but cannot be used for reports to the player because such reports would be based on (supposedly nonexistent) operational accounting. The two classes of states are listed in tables 2 and 3. Table 2 contains the external states that can be reported to players; table 3 lists the internal states that cannot. All inventory taking entails costs, except the cash inventory, which is free. Inventory data can also be computed independently from event data. Computing an inventory both ways is meaningful for control only; otherwise a preference for inventory taking over accounting from event data will be motivated by cost.

Table 2: External states

State	Explanation
Cash	Amount of cash or bank balance.
Materials	Amount in stock for each type of material, excluding materials allocated to current production orders.
Products	Finished products in stock for each type of product.
Operating machines	Machines in operation, classified by machine type and job number. Individual machines are not identified.
Idle machines	Machines currently not in use, classified by machine type, and divided into usable machines and machines in repair. Individual machines are not identified.
Employees	Total number of employees.
Operators	Number of operators classified by job number.

Table 3: Internal states

State	Explanation
Consumer orders	A list of orders from consumers, characterized by product name, consumer number, amount, price and latest delivery date.
Production orders	A list of orders to produce a product, characterized by product name, amount and latest delivery date.
EconStock of materials	Economic stock; the sum of the real stock from table 2 and the amount ordered from suppliers.
EconStock products	Economic stock; the real stock from table 2 plus the amount to be produces by jobs in process minus the amount ordered by consumers. The result can be negative.
Productivity	Average productivity of employees.

4 IMPLEMENTATION

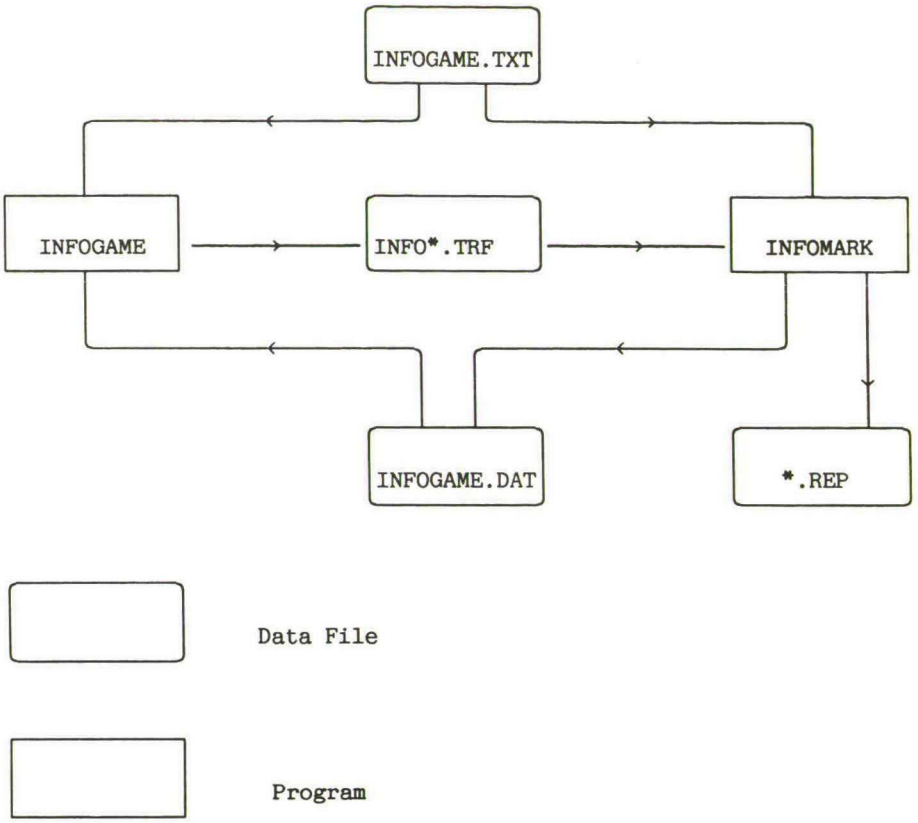
4.1 Programming language and environment

Pascal was chosen as the implementation language for Infogame because we considered it the best language for which implementations are currently available on a wide range of computers. The program has been developed in Turbo-Pascal on an IBM-PC. Initially, Rev 3 was used; in the spring of 1988, we switched to Rev 4, mainly to profit from the increased modularity offered by its units. Even porting between compiler versions asks for a fair number of changes, but such changes are minor when compared to changes in specifications. For example, when a gaming program is ported from a microcomputer to a mainframe or minicomputer, it is a major task to allow simultaneous input from players, which is not possible on a standard PC. We would have preferred a language with better facilities for the definition of abstract data types, such as SIMULA, but the main characteristics of Pascal, (its procedurality and the prevalence of memory assignment) are not seen as disadvantages. Limitations in memory space and processing speed were seen as a reminder to keep it simple [15]. The resulting program is quite small: it contains some 4300 program lines or 3400 Pascal statements.

4.2 Program structure

The Infogame system contains two programs: INFOGAME accepts player inputs; INFOMARK simulates the actual operation of companies and markets. Data on the initial state of the simulated world are prepared by the game administrator and read by both programs; data on state changes are in a file that is written by INFOMARK and can also be read by INFOGAME. Data entered in INFOGAME are stored in an intermediate file that can be changed until INFOMARK is started for the next round. INFOMARK retrieves all input data from this intermediate file but does not write to it. INFOMARK writes output to a report file for further processing by the player.The relation between files and programs is shown in Fig. 14.

Fig. 14: Files and programs



The primary reason to split the program into two parts was its size, but dividing the game program into an input part and a simulation part has some additional advantages. First, the input part can be changed without changing the simulation program as long as it delivers an intermediate file in the same format. In this way, tests were made with a simple version of INFOGAME with default values for many decisions. It is also possible to replace INFOGAME with an intelligent program that computes decisions from results in the last round. Second, the well-defined interface between the two program parts simplifies testing.

4.3 Input data

For input, three mechanisms are used:

Selection: An option is selected from a list, and the programs proceeds with the instructions pertaining to that option.

Set selection: A set of options is selected from a list; pressing the end key signals that the next instruction should be executed.

Table filling: A table that can be filled in WYSIWYG fashion is presented on the screen. Input data are read and checked after pressing the end key has signalled the end of input.

Selection and table filling are present in almost any software package. Set selection is less common because it can be used only for options that are nodes in the menu tree.

With selection, input errors are not possible, because only correct options are shown on the screen (a management game is not a multiple-choice examination). With set selection, input errors are possible in theory, but they cannot occur in Infogame. Consequently, input errors in Infogame are made only during table filling.

We distinguish two types of input errors, clerical errors and errors in judgment. The input checker in a management game should check on clerical errors, such as misspelling the name of a supplier, but it should not signal errors in judgment, such as setting a price that is too low or too high. Not all errors can, however, be classified as clearly. For example, trying to start production without acquiring the necessary resources can be considered a clerical error, because an analysis of input data will accurately predict a zero production rate, but the cause of this error may be a fundamental misconception of

the simulated system. On the other hand, misstating a number by a factor 10 is a clerical error by nature, but it is hard to detect for an input checker. For a realistic simulation of organizations, we should ask whether an instruction would be questioned in a real-world environment. This depends on the structure of the organization and the style of leadership. Under authoritarian leadership, instructions will not be questioned, but their consequences will be circumvented. For example, salesmen who cannot compete in the market will exploit the full gamut of discounts instead of telling their authoritarian marketing manager that list prices are too high. Under democratic leadership, instructions will be questioned, and they will be corrected when in error. Democracy is naturally implemented when a game is played by team, with team members checking decisions. If we consider the input program as a middle manager, she expects authoritarian behaviour from the player or player team: input is checked for syntax and completeness only. This approach was chosen for practical reasons. First, corrections or warnings from the input program cannot be used if decisions are written down on forms and entered afterwards. Second, input checking demands a fairly high amount of intelligence in the input program and we did not think this effort our first priority.

The main task of the input program is to convert screen data to the intermediate data file. Its format is described in section 4.4 to enable users to write their own input preparation program. Such a program can be written without knowledge of the internal data structures that INFOGAME shares with INFOMARK, because these are used only for temporary storage and for retrieving input values from preceding periods.

4.4 The intermediate file

The intermediate data file prepared by INFOGAME and used by INFOMARK is named INFOna.TRF, where n is a one- or two-digit number indicating the period and a is a sequence letter from 'A' to 'L'. Each *.TRF file contains input data from a single firm; sequence letters are assigned sequentially. For example, a player who is the first to start input in period 5, is assigned the file INF05A.TRF. Separate input files were specially introduced to allow simultaneous input in a network with shared files. Each input file is an ASCII file consisting of 13 blocks of 12 lines each. Most lines will be empty, but little space is wasted in this way, as empty lines are represented by single return characters in the *.TRF file and by null pointers in memory. Each block is interpreted differently (this is another reason to use the versatile ASCII format). The lines in the block can be structured in three ways: (1) only the first line is used, (2) the first and the second line are interpreted differently, and (3) all lines are interpreted in the same way.

Block 1: Company data, line 1

1- 8 Company name

9-15 Password

Block 2: Reports, line 1

1-60 '0' Event or state not reported

'1' Event reported

Line 2:

1- 3 Number of days between reports on the first state actually reported.

4- 7 Number of days between reports for second state

....

Block 3: Investment, all lines

1- 8 Name of machine type

9-12 Number of machines

Block 4: Scrap machines, all lines

1- 4 Number of n th machine to be scrapped, where n = 20(i-1)+1 if i is the number of the line within this block.

.... Number of next machine

Block 5: Design a product, all lines

1- 8 Product name

9-15 Name of first process used

.... Name of next process (maximal number of process names = 4)

Block 6: Remove a product, all lines

1-8 Name of n th product to be removed, where n = 10(i-1)+1 if i is the number of the line within this block

.... Name of next product

Block 7: Define product variables, all lines

1- 8 Product name

9-14 Delivery time

15-24 Reorder level

25-34 Order quantity

35-40 Maximum time

41-46 Price

47-58 Advertising budget

Block 8: Variables for materials, all lines

1- 6 Quality index

7-16 Reorder level

17-26 Order quantity

Block 9: Suppliers, all lines

1- 6	Quality index of material
7-10	Number of suppliers
11-18	Name of first supplier
.....	Name of next supplier (maximum 4 suppliers)

Block 10: Employee data, line 1

1- 6	Term after which employee may be fired
7-16	Personnel advertising budget
17-22	Planned to hire
23-28	Planned to fire
29-38	Salary level

Block 11: Bank loans, line 1

1-12	Limit to overdraft agreed
13-24	Long term loan agreed
25-30	Interest percentage for long term loan
31-36	Duration of long term loan

Block 12: Repay bank loan, all lines

1-12	Amount of loan to be repaid
13-18	Duration of loan
19-24	Interest percentage of loan

Block 13: Acceptance and granting of credit, line 1

1-12	'0' Credit from this supplier is not accepted '1' Credit from this supplier is accepted

Line 2

1- 6	Days credit granted in industry nr 0.
....	Days credit in next industry. Normally, there are 5 industries, from which industries 1-4 are consumer industries, where credit can be granted.

4.5 Data structures

Conceptually, the data structure in Infogame can be represented in the entity-relationship model [9,22]. Entities are either concrete entities such as machines, or abstractions such as machine types. Examples of relationships are "use" (a technology uses a type of machine) and "assigned to" (a machine is assigned to a job). Conceptual schema design for a management game or simulation program differs from conceptual design for an information system in the real world because the designer can choose the level of abstraction. For example, an earlier version of Infogame incorporated the notion of a "production line", to which machines and workers were assigned semi-permanently. In the present version, machines and workers are assigned to single jobs. In information systems design, such a change can occur only as a result of changes in production organization, and not because it simplifies system design. Another difference is that algorithms, for example scheduling algorithms, are an essential feature of management game programs, whereas the corresponding decisions are treated as external events by the majority of information systems. So management games are not suitable for formal methods deriving information systems from conceptual schemata such as [21]. The conceptual schema is shown in Fig. 15A through Fig. 15C; connectivity is represented as in [22].

Physically, Infogame uses two representations, a file representation and an internal memory representation. In this way, management games differ from conventional information systems, where all data is stored in a DBMS, and from simple simulation programs, that store all data in memory. This is because, on the one hand, extensive use in computations demands fast access to data, but on the other hand, disk backup between playing rounds is necessary as the stakes in management gaming are too high to risk loss of data as a result of power failures or operator errors. Disk backup could be provided by simply dumping memory contents, but their are other uses for file storage, such as restarting a previous round and intermittent play. For files, the most important requirement is economy in disk use.

Fig. 15A: Relations to job

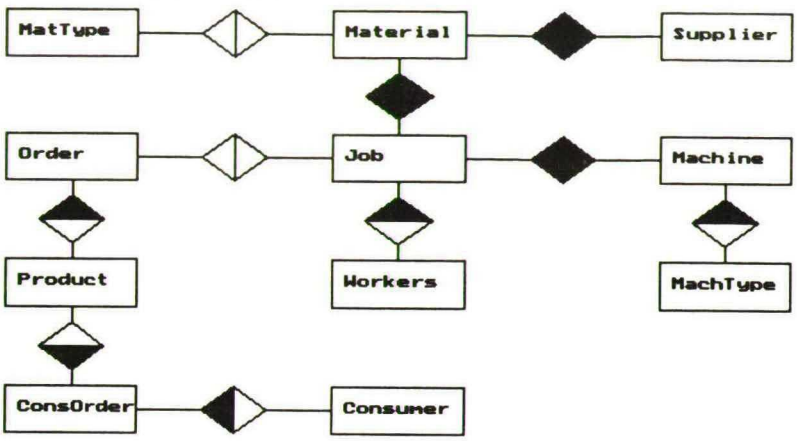


Fig. 15B: Relations to company

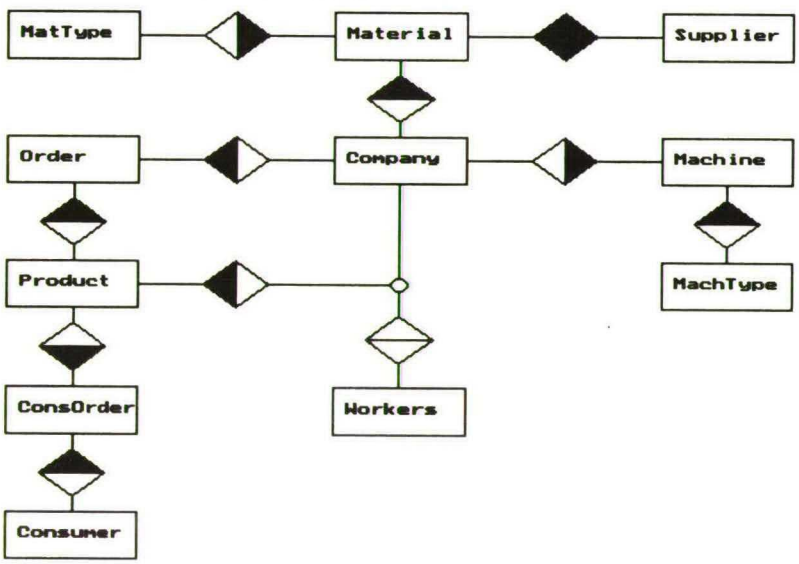
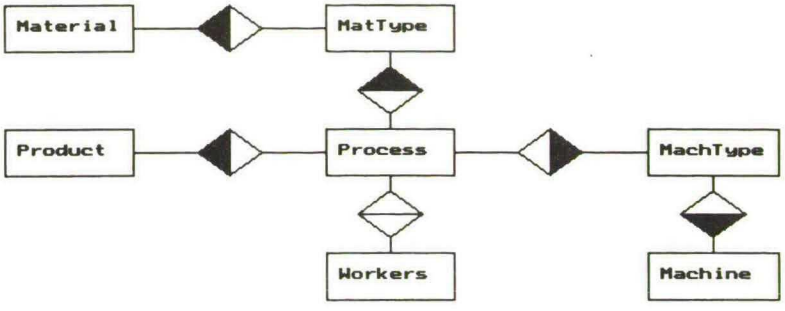


Fig. 15C: Relations to process (technology)



For the memory representation, there is an upper limit to the amount of memory available, and Turbo-Pascal has the additional peculiarity that the major part of memory is only accessible through pointers, but otherwise, speed and simplicity of data access are the prime requirements.

In the design process, the memory representation was defined first; the file representation was derived from it. To represent sets of entities, such as machines or products, we had to choose between two well-known representations: the linked list and the array [25]. Linked lists offer easy insertion and deletion, arrays are more suitable for direct access by index value. The argument that lists in Turbo-Pascal are accessed by pointers and thus use the more abundant heap memory is not valid, because arrays can use heap memory as well.

Machine types, technologies, and suppliers are represented by arrays. Attribute data for these entities are read from a textfile before the start of each round. Originally, we thought it should be possible to add and delete machine types, technologies, and suppliers during play; accordingly we represented those entities by a pointer for internal use and by name for external use. Subsequently, the array structure was chosen for efficiency. This change would not have been necessary if Pascal supported a mechanism for direct addressing by name, as in Database languages, and in programming languages like ICON [11] and B [18]. The set of types of material is not represented by a data structure at all, as its nominal quality, which is used as a name, is the only attribute of a type of material.

Firms, consumers and employees are also represented by arrays. Attribute values for these entities may change during a round; consequently, those values are stored in a datafile at the end of a round and retrieved at the start of the next round. New firms can be added in any round, but once a firm is removed, it retains its number, even if it stops activity.

The number of consumers remains constant; growth in consumer expenditure is attained by a decrease in interarrival time for existing consumers and an increase in the average size of consumer orders. Potential employees can be added as well as removed. To prevent problems with references to employee numbers, some restrictions have been introduced; for example, no employee will leave the labor market when changing jobs. The data structure for each firm must also efficiently store all job applications. As the number of applicants may be equal to the number of potential employees (currently 8000), and Turbo-Pascal supports neither one-bit Boolean arrays nor large sets, an array of sets is used.

Machines, materials, products, consumerorders, production orders, and loans are represented as linked lists. This representation is essential for production orders, where the list is frequently searched for executable jobs.

A product is linked both to the next product in the firm (for the production model), and to the next product in the industry (for the marketing model). Because data are stored on file after each round, there is no need for garbage collection.

The event list is structured as a heap, which is a simple method guaranteeing O(n log N) storage time for events [14]. Event records (with variants for different types of events) are stored on the Pascal heap; pointers to event records are manipulated in an array in global memory. Space allocated to the event list is reused when possible. Once entered in the event list, an event is never deleted before the planned time; this has influenced game design in some details: for example, ending time of a job is never recomputed once it has been started. As the event list may contain events occuring after the end of a period it is stored on file at the end of a period and read from file at the start of a period.

Data structures on file are similar to the corresponding data structures in memory. However, both linked lists and arrays are represented as sequential files, and all pointers in memory are replaced by appropriate identifiers in the file. The major part of data is stored in INFOGAME.DAT; the record type of this file contains variant types for each record in memory that is stored. For efficient use of disk space, some variants contain arrays of records rather than single records. Data on potential employees, consumers, and events are stored in separate files. This was originally done to save disk space. It also saves reading tinme in INFOGAME, because these files are only used in INFOMARK.

Unlike relational databases and conceptual schemata, programming languages like Pascal cannot directly represent a relationship in a symmetric fashion. As an example, we show the representation of the fact that firm "ABC" produces products "A" and "B".

In a relational database, this is represented by the relation:

PRODUCTS

Company	Product
ABC	A
ABC	B

From this relation we can find the producer of "A" with

```
SELECT Company FROM PRODUCTS WHERE Product = "A"
```

We can also find all products produced by "ABC" with:

```
SELECT Product FROM PRODUCTS WHERE Company = "ABC"
```

In a Pascal-like language such a relation can be implemented by a reference (which can be represented by a Pascal pointer, an index number, or a name) to the company in every product record, by a list or array with references to products in every company record, or both ways. If there are n companies, and m products per company, finding all products of a company takes O(nm) time if the relation is represented only by company references in the product record, and O(m) time if company records contain lists of products. On the other hand, retrieving the company producing a product takes O(1) time if the product record refers to the company, and O(nm) time otherwise. Storage requirements for both representations have O(nm) complexity.

If the two operations, to find the products of a company, and to find the company producing a product, are executed frequently, using both representations at the same time reduces computation time. The main disadvantage of this approach is not the additional demand on memory, but the addition of an invariant, stating that a product contains a pointer to a company if and only if the list of products of that company contains a pointer to that product.

Such an invariant can be violated with any addition or deletion of a relevant entity, and with any change in the corresponding relationship. As the concern for computation speed diminished during program development, we traded speed for security by removing some of the redundant references from the program.

When choosing a representation for references, we should consider that the operations to retrieve an element from an array and to retrieve a data item with a pointer, use generic operations [] and † respectively, whereas searching a list for a name requires a specialized function. On the other hand, both names and indices can be stored in external files, whereas pointers cannot. Thus, as known by Fortran programmers from the late fifties, the primitive array structure is often the most efficient.

References

1 Anthonisse,J.M., K.M. Van Hee and J.K. Lenstra: Resource-constrained project scheduling: an international exercise in DSS development. CWI, Amsterdam, 1987.

2 Boehm,B.W., T.E.Gray and T.Seewaldt: Prototyping versus specifying: a multiproject experiment. IEEE Transactions on Software Engineering, Vol SE-10 (May 1984), p 290-303.

3 Brinch Hansen, P.: Operating system principles. Prentice Hall, Englewood Cliffs N.J., 1973.

4 Buffa,E.S.: Modern production/operations management (7th ed.). John Wiley & sons, New York, 1983.

5 Casimir,R.J.: DSS, information systems, and management games. Information and Management, Vol. 11, No 3 (Oct. 1986). pp 123-129.

6 Casimir,R.J.: Infogame, the model. Reeks "ter discussie" 87.07, Tilburg University, Tilburg, 1987.

7 Casimir,R.J: Market models in management games. in : Ontwerpen van bedrijfsspelen (design of management games). Reeks "ter discussie" 87.09, Tilburg University, Tilburg, 1987.

8 Casimir,R.J.: Infogame users manual. Research Memorandum FEW 363, Tilburg University, Tilburg, 1988.

9 Chen,P.P.: The entity-relationship model-toward a unified view of data. TODS Vol. 1 no. 1 (March 1976), pp. 9-36.

10 Fox, B.R. and K.G. Kempf: Complexity, uncertainty and opportunistic scheduling. in: Weisbin, C.R. (ed.): Artificial intelligence applications, the engineering of knowledge-based systems. North Holland, Amsterdam 1985.

11 Griswold,R.E. and M.E. Griswold: The ICON programming language Prentice-Hall, Englewood Cliffs, N.J., 1983.

12 Gross,D. and C.M.Harris: Fundamentals of queuing theory. John Wiley & sons, New York, 1985.

13 Holt,C.C.: Improving the labor market trade-off between inflation and unemployment. Am. Ec. Review Vol 59 No 2 (1969), p. 134-146.

14 Jones,D.W.: An empirical comparison of priority-queue and event-set implementations. CACM Vol. 29 no. 4 (april 1986), pp 300-311.

15 Kleijnen, J.P.C.: Personal communication.

16 Kotler, P.: Principles of marketing, 3d ed.
Prentice-Hall, Englewood Cliffs, 1986.

17 Lundeberg,M., G.Goldkuhl and A.Nilsson: Information system development, a systematic approach. Prentice-Hall, Englewood Cliffs,N.J., 1981.

18 Meertens,L.: Draft proposal for the B programming language. Centre for Mathematics, Amsterdam, 1981.

19 Shaw, M.J. and A.B. Whinston: Task bidding and distributed planning in flexible manufacturing.
in: Weisbin, C.R. (ed.): Artificial intelligence applications, the engineering of knowledge-based systems.
North Holland, Amsterdam 1985.

20 Schroeder, R.G.: Operations management.
McGraw-Hill, New York, 1981

21 Troyer,O. de, R.Meersman and P.Verlinden: RIDL on the CRIS case: a workbench for NIAM. Tilburg University, Tilburg, 1988.

22 Teory. T.J., D.Yang and J.P. Fry: A logical design methodology for relational databases using the extended entity-relationship model. Computing Surveys Vol 18 No2 (June 1986), p. 197-222.

23 Van Schaik,F.D.J.: Effectiveness of decision support systems. Delft University Press, Delft, 1988.

24 Winters,A and E.Hendrix: Het reduceren van de simulatie in een productie-voorraad systeem. (Simulation reduction in a production-stock system) Unpublished research memorandum, Tilburg University, Tilburg, 1986.

25 Wirth,N. Algorithms + data structures = programs. Prentice Hall, Englewood Cliffs, N.J., 1976.

26 Zimmermann, H.J. and Sovereign, M.G.: Quantitative models for production management. Prentice Hall, Englewood Cliffs, 1974.

IN 1988 REEDS VERSCHENEN

297 Bert Bettonvil
Factor screening by sequential bifurcation

298 Robert P. Gilles
On perfect competition in an economy with a coalitional structure

299 Willem Selen, Ruud M. Heuts
Capacitated Lot-Size Production Planning in Process Industry

300 J. Kriens, J.Th. van Lieshout
Notes on the Markowitz portfolio selection method

301 Bert Bettonvil, Jack P.C. Kleijnen
Measurement scales and resolution IV designs: a note

302 Theo Nijman, Marno Verbeek
Estimation of time dependent parameters in lineair models
using cross sections, panels or both

303 Raymond H.J.M. Gradus
A differential game between government and firms: a non-cooperative approach

304 Leo W.G. Strijbosch, Ronald J.M.M. Does
Comparison of bias-reducing methods for estimating the parameter in dilution series

305 Drs. W.J. Reijnders, Drs. W.F. Verstappen
Strategische bespiegelingen betreffende het Nederlandse kwaliteits-concept

306 J.P.C. Kleijnen, J. Kriens, H. Timmermans and H. Van den Wildenberg
Regression sampling in statistical auditing

307 Isolde Woittiez, Arie Kapteyn
A Model of Job Choice, Labour Supply and Wages

308 Jack P.C. Kleijnen
Simulation and optimization in production planning: A case study

309 Robert P. Gilles and Pieter H.M. Ruys
Relational constraints in coalition formation

310 Drs. H. Leo Theuns
Determinanten van de vraag naar vakantiereizen: een verkenning van materiële en immateriële factoren

311 Peter M. Kort
Dynamic Firm Behaviour within an Uncertain Environment

312 J.P.C. Blanc
A numerical approach to cyclic-service queueing models

313 Drs. N.J. de Beer, Drs. A.M. van Nunen, Drs. M.O. Nijkamp
Does Morkmon Matter?

314 Th. van de Klundert
Wage differentials and employment in a two-sector model with a dual labour market

315 Aart de Zeeuw, Fons Groot, Cees Withagen
On Credible Optimal Tax Rate Policies

316 Christian B. Mulder
Wage moderating effects of corporatism
Decentralized versus centralized wage setting in a union, firm, government context

317 Jörg Glombowski, Michael Krüger
A short-period Goodwin growth cycle

318 Theo Nijman, Marno Verbeek, Arthur van Soest
The optimal design of rotating panels in a simple analysis of variance model

319 Drs. S.V. Hannema, Drs. P.A.M. Versteijne
De toepassing en toekomst van public private partnership's bij de grote en middelgrote Nederlandse gemeenten

320 Th. van de Klundert
Wage Rigidity, Capital Accumulation and Unemployment in a Small Open Economy

321 M.H.C. Paardekooper
An upper and a lower bound for the distance of a manifold to a nearby point

322 Th. ten Raa, F. van der Ploeg
A statistical approach to the problem of negatives in input-output analysis

323 P. Kooreman
Household Labor Force Participation as a Cooperative Game; an Empirical Model

324 A.B.T.M. van Schaik
Persistent Unemployment and Long Run Growth

325 Dr. F.W.M. Boekema, Drs. L.A.G. Oerlemans
De lokale produktiestructuur doorgelicht.
Bedrijfstakverkenningen ten behoeve van regionaal-economisch onderzoek

326 J.P.C. Kleijnen, J. Kriens, M.C.H.M. Lafleur, J.H.F. Pardoel
Sampling for quality inspection and correction: AOQL performance criteria

327 Theo E. Nijman, Mark F.J. Steel
Exclusion restrictions in instrumental variables equations

328 B.B. van der Genugten
Estimation in linear regression under the presence of heteroskedasticity of a completely unknown form

329 Raymond H.J.M. Gradus
The employment policy of government: to create jobs or to let them create?

330 Hans Kremers, Dolf Talman
Solving the nonlinear complementarity problem with lower and upper bounds

331 Antoon van den Elzen
Interpretation and generalization of the Lemke-Howson algorithm

332 Jack P.C. Kleijnen
Analyzing simulation experiments with common random numbers, part II: Rao's approach

333 Jacek Osiewalski
Posterior and Predictive Densities for Nonlinear Regression.
A Partly Linear Model Case

334 A.H. van den Elzen, A.J.J. Talman
A procedure for finding Nash equilibria in bi-matrix games

335 Arthur van Soest
Minimum wage rates and unemployment in The Netherlands

336 Arthur van Soest, Peter Kooreman, Arie Kapteyn
Coherent specification of demand systems with corner solutions and endogenous regimes

337 Dr. F.W.M. Boekema, Drs. L.A.G. Oerlemans
De lokale produktiestruktuur doorgelicht II. Bedrijfstakverkenningen ten behoeve van regionaal-economisch onderzoek. De zeescheepsnieuwbouwindustrie

338 Gerard J. van den Berg
Search behaviour, transitions to nonparticipation and the duration of unemployment

339 W.J.H. Groenendaal and J.W.A. Vingerhoets
The new cocoa-agreement analysed

340 Drs. F.G. van den Heuvel, Drs. M.P.H. de Vor
Kwantificering van ombuigen en bezuinigen op collectieve uitgaven 1977-1990

341 Pieter J.F.G. Meulendijks
An exercise in welfare economics (III)

342 W.J. Selen and R.M. Heuts
A modified priority index for Günther's lot-sizing heuristic under capacitated single stage production

343 Linda J. Mittermaier, Willem J. Selen, Jeri B. Waggoner, Wallace R. Wood
Accounting estimates as cost inputs to logistics models

344 Remy L. de Jong, Rashid I. Al Layla, Willem J. Selen
Alternative water management scenarios for Saudi Arabia

345 W.J. Selen and R.M. Heuts
Capacitated Single Stage Production Planning with Storage Constraints and Sequence-Dependent Setup Times

346 Peter Kort
The Flexible Accelerator Mechanism in a Financial Adjustment Cost Model

347 W.J. Reijnders en W.F. Verstappen
De toenemende importantie van het verticale marketing systeem

348 P.C. van Batenburg en J. Kriens
E.O.Q.L. - A revised and improved version of A.O.Q.L.

349 Drs. W.P.C. van den Nieuwenhof
Multinationalisatie en coördinatie
De internationale strategie van Nederlandse ondernemingen nader beschouwd

350 K.A. Bubshait, W.J. Selen
Estimation of the relationship between project attributes and the implementation of engineering management tools

351 M.P. Tummers, I. Woittiez
A simultaneous wage and labour supply model with hours restrictions

352 Marco Versteijne
Measuring the effectiveness of advertising in a positioning context with multi dimensional scaling techniques

353 Dr. F. Boekema, Drs. L. Oerlemans
Innovatie en stedelijke economische ontwikkeling

354 J.M. Schumacher
Discrete events: perspectives from system theory

355 F.C. Bussemaker, W.H. Haemers, R. Mathon and H.A. Wilbrink
A (49,16,3,6) strongly regular graph does not exist

356 Drs. J.C. Caanen
Tien jaar inflatieneutrale belastingheffing door middel van vermogensaftrek en voorraadaftrek: een kwantitatieve benadering

357 R.M. Heuts, M. Bronckers
A modified coordinated reorder procedure under aggregate investment and service constraints using optimal policy surfaces

358 B.B. van der Genugten
Linear time-invariant filters of infinite order for non-stationary processes

359 J.C. Engwerda
LQ-problem: the discrete-time time-varying case

360 Shan-Hwei Nienhuys-Cheng
Constraints in binary semantical networks

361 A.B.T.M. van Schaik
Interregional Propagation of Inflationary Shocks

362 F.C. Drost
How to define UMVU

363 Rommert J. Casimir
Infogame users manual
Rev 1.2 December 1988

364 M.H.C. Paardekooper
A quadratically convergent parallel Jacobi-process for diagonal dominant matrices with nondistinct eigenvalues

365 Robert P. Gilles, Pieter H.M. Ruys
Characterization of Economic Agents in Arbitrary Communication Structures

366 Harry H. Tigelaar
Informative sampling in a multivariate linear system disturbed by moving average noise

367 Jörg Glombowski
Cyclical interactions of politics and economics in an abstract capitalist economy

IN 1989 REEDS VERSCHENEN

368 Ed Nijssen, Will Reijnders
"Macht als strategisch en tactisch marketinginstrument binnen de distributieketen"

369 Raymond Gradus
Optimal dynamic taxation with respect to firms

370 Theo Nijman
The optimal choice of controls and pre-experimental observations

371 Robert P. Gilles, Pieter H.M. Ruys
Relational constraints in coalition formation

372 F.A. van der Duyn Schouten, S.G. Vanneste
Analysis and computation of (n,N)-strategies for maintenance of a two-component system

373 Drs. R. Hamers, Drs. P. Verstappen
Het company ranking model: a means for evaluating the competition